Constrained static optimization in economics: methods, algorithms and implementation in the general algebraic modeling system

Germán D. Lambardi and P. Ruben Mercado

Uade

July 2002

Optimización estática restringida en economía: métodos, algoritmos e implementación en el general algebraic modeling system

Julio 2002

Lic. Germán Lambardi y Dr. P. Ruben Mercado

Resumen

Se presentan métodos matemáticos de programación matemática lineal y no lineal, y su implementación computacional en GAMS (General Algebraic Modeling System). Se presentan ejemplificaciones económicas y se introducen diversos algoritmos de solución: simplex, gradiente, de Newton y de penalidades.

Abstract

The paper presents methods of linear and nonlinear mathematical programming and their computational implementation in the General Algebraic Modeling System (GAMS). It also presents economic examples and introduces a number of solution algorithms: simplex, gradient, Newton and penalties.

Palabras clave: optimización estática, programación lineal, programación no lineal, GAMS

Keywords: static optimization, linear programming, nonlinear programming, GAMS

JEL classification: C6 Mathematical Methods; Programming Models; Mathematical and Simulation Modeling

Programa de Investigación en Economía Computacional
DEyF -UADE
Lima 717 Piso 1
(1073) Buenos Aires
Argentina

El problema general de la optimización estática restringida es el de maximizar o minimizar una función objetivo sujeta a restricciones.[1] Cuando éstas restricciones toman la forma de igualdades, el problema suele resolverse apelando a los métodos clásicos del análisis matemático. Si las mismas toman la forma de desigualdades, es necesario recurrir a lo que se conoce como programación matemática.

Si bien existen dentro de la programación matemática ciertos resultados analíticos que permiten especificar condiciones necesarias y suficientes para la solución de un problema dado, usualmente es necesario recurrir a métodos numéricos basados en algoritmos específicos. En tal sentido, para problemas de programación lineal -donde la función objetivo y las restricciones son todas lineales- se aplica el método Simplex, mientras que en el caso de problemas de programación no lineal -donde la función objetivo y/o las restricciones pueden ser no lineales- se suelen aplicar los que se conocen como métodos gradiente, de Newton y de "penalidades".

El General Algebraic Modeling System (GAMS) es un software de alto nivel de uso extendido en aplicaciones económicas. El mismo permite representar modelos y problemas de optimización o simulación en una forma muy modular e independientemente del método numérico específico utilizado para resolverlos, el cual puede ser elegido por el usuario entre una batería de "solvers" disponible.

En lo que sigue, presentaremos una introducción a los métodos y algoritmos más usuales de programación lineal y no lineal, y a la implementación de problemas en GAMS. En la Sección 1 nos concentraremos en problemas de programación lineal, mientras que en la Sección 2 abordaremos problemas de programación no lineal.

1.1) Programación Lineal:[2]

La programación lineal trata con problemas en los que se debe maximizar o minimizar una función lineal sujeta a un sistema de igualdades/desigualdades lineales. Usualmente, en aplicaciones económicas se requiere adicionalmente una condición de no negatividad de las variables. Formalmente, el problema puede expresarse como el hallar los valores de las variables x_j *(j= 1, ..., r)* que maximizan:

$$z = \sum_{j=1}^{r} p_j x_j \tag{1.1}$$

sujeto a:

$$\begin{gathered} a_{11}x_1 + \cdots + a_{1r}x_r \leq b_1 \\ \vdots \\ a_{m1}x_1 + \cdots + a_{mr}x_r \leq b_m \\ x_1, \ldots, x_r \geq 0 \end{gathered} \tag{1.2}$$

[1] En Mercado (2002) se trata con problemas de optimización dinámica restringida y su implementación en GAMS.

[2] Para un tratamiento extensivo ver, Chiang (1987), Silberberg et. al. (2001) y Rardin (1998).

donde p_j, a_{ij} y b_i pueden se positivos, negativos o nulos. Cuando se trata de minimizar una función lineal, el problema puede rescribirse utilizando el formato (1.1)-(1.2) maximizando la función con signo negativo. Asimismo, si una restricción es de la forma $\geq$, la desigualdad puede invertirse multiplicando a ambos lados por –1.

Cualquier conjunto de números reales que satisfaga las desigualdades del problema anteriormente planteado, constituye una solución factible del sistema. Todas las soluciones factibles conforman el conjunto factible. Por ejemplo, si tenemos el clásico problema de análisis de actividad, donde hay que maximizar el beneficio bruto de la venta de dos productos cuyos precios son iguales a \$2 y \$3 respectivamente, sujeto a restricciones físicas sobre la producción factible de la empresa, el mismo puede formalizarse como el de maximizar:

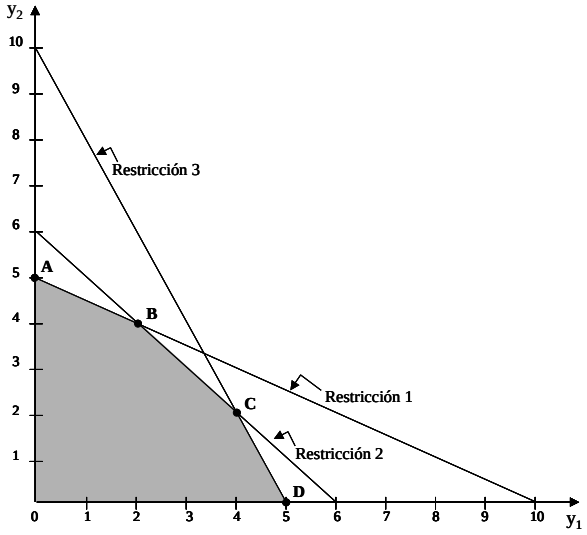
$$z = 2y_1 + 3y_2 \tag{1.3}$$

sujeto a:

restricción 1)	$0.5y_1 + y_2 \leq 5$	(1.4)
restricción 2)	$y_1 + y_2 \leq 6$	(1.5)
restricción 3)	$2y_1 + y_2 \leq 10$	(1.6)
restricción de no negatividad	$y_1, y_2 \geq 0$.	(1.7)

La representación del conjunto factible correspondiente viene dada por el siguiente gráfico:

Grafico 1



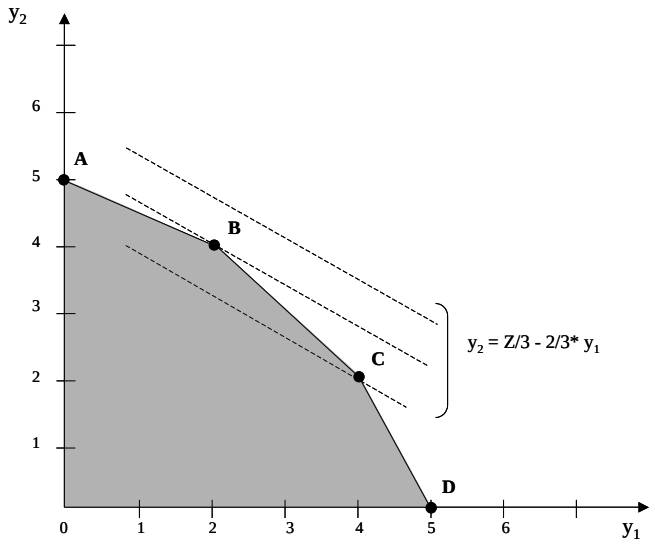
En todos los problemas de programación lineal del formato (1.1)-(1.2) la región factible tiene la propiedad de ser un conjunto convexo, cerrado e inferiormente acotado. Esto, aunado a que la función objetivo es lineal, nos permite afirmar que si un máximo de esta función es alcanzado sobre la región factible y si el mismo único, se trata de un máximo global. Si dicho máximo no es único, tendremos un numero infinito de máximos, hallándose estos sobre el segmento que une dos máximos cualesquiera.

Una vez definido el conjunto factible se trata de hallar el punto (y_1, y_2) de dicho conjunto que maximiza el valor de la función objetivo, en nuestro caso $z = 2y_1 + 3y_2$. Para nuestro ejemplo podemos graficar (Gráfico 2) las curvas de nivel de la función *z*, es de decir las combinaciones de y_1 e y_2 que permiten alcanzar un determinado nivel de *z*. Dichas curvas de nivel son todas paralelas y su ecuación es:

$$y_2 = \frac{z^0}{3} - \frac{2}{3} y_1 . \qquad (1.8)$$

donde z^0 es el valor de la función objetivo elegido. A medida que nos alejamos del origen de coordenadas, las curvas de nivel tienen asociado un valor más elevado de *z*. La solución a nuestro problema es encontrar la combinación de (y_1, y_2) que nos permita alcanzar la curva de nivel más alejada, lo cual se da en el punto B.

Grafico 2



El punto B resulta de la intersección de la restricción (1.4) con la restricción (1.5). Resolviendo el sistema obtenemos $y_1^* = 2, y_2^* = 4$, lo cual implica que z^0 = 16. Podemos observar en el Gráfico 2 que el punto B queda por debajo de la restricción $2y_1 + y_2 \leq 10$, lo que implica que de hecho sobran 10-2(2)-(4) = 2 unidades del recurso, es decir que existe una "holgura".

Si bien el método gráfico deja de ser aplicable a medida que tenemos más variables en el problema, el mismo nos proporciona dos intuiciones básicas. La primera es que si existe solución al problema, esta se encontrará en alguno de los vértices del conjunto factible, en nuestro ejemplo los puntos A, B, C y D. La segunda es que puede existir más de

una solución al problema cuando las curvas de nivel son paralelas a alguna de las restricciones. Esto último se pude observar claramente si, por ejemplo, la función objetivo hubiera sido $z = 2y_1 + 2y_2$, donde las curvas de nivel tendrían pendiente -1, al igual que la restricción (1.5). En este caso habría infinitas soluciones: todos los puntos que unen los vértices B y C. Nótese que también los vértices B y C serían parte de la solución.[3]

Estos resultados reducen drásticamente el número puntos factibles sobre los cuales debemos buscar una solución. De esta manera, un algoritmo para la solución de nuestro problema podría consistir simplemente en la evaluación de la función objetivo en cada uno de los "vértices" , escogiéndose aquel que nos proporcione el valor de *z* más elevado. Sin embargo, este método es altamente ineficiente ya que a medida que aumenta el tamaño de nuestro problema el numero de "vértices" a ser evaluados crece en forma explosiva. Por ejemplo, un modelo de 10 ecuaciones y 20 variables puede contener hasta:

$$\binom{20}{10} = \frac{20!}{10!10!} = 184.756 .$$

"vértices".[4]

Un método más eficiente es el método Simplex, el cual nos "guía" en la búsqueda de los vértices que constituyen una solución al problema[5]. El primer paso de éste método consiste en convertir las restricciones de desigualdad en restricciones de igualdad adicionando las variables de "holgura". De esta manera, nuestro problema queda formulado como el de maximizar:

$$z = 2y_1 + 3y_2 \tag{1.9}$$

sujeto a:

$$0.5y_1 + y_2 + y_3 = 5 \tag{1.10}$$

$$y_1 + y_2 + y_4 = 6 \tag{1.11}$$

$$2y_1 + y_2 + y_5 = 10 \tag{1.12}$$

$$y_1, y_2, y_3, y_4, y_5 \geq 0 . \tag{1.13}$$

Las variables de holgura están sujetas a la restricción de no ser negativas, por lo tanto la zona factible formada por las restricciones de igualdad es la misma región factible que antes, de tal modo que el nuevo problema es equivalente al anterior. Ahora tenemos tres restricciones independientes y cinco variables. En notación vectorial, nuestras restricciones pueden expresarse como:

---

[3] Las "intuiciones" presentadas pueden demostrarse matemáticamente en forma rigurosa. Ver Chiang (1987), Silberberg et al. (2001) y Simon et al. (1994).

[4] Ver Silberberg (2001), Capítulo 17.

[5] Nuestro objetivo no es presentar una visión general de este procedimiento. Para una visión de la mecánica del método Simplex ver Chiang (1987), Capítulo 19.

$$\begin{bmatrix} 0.5 \\ 1 \\ 2 \end{bmatrix} y_1 + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} y_2 + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} y_3 + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} y_4 + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} y_5 = \begin{bmatrix} 5 \\ 6 \\ 10 \end{bmatrix}. \qquad (1.14)$$

Si igualamos a cero dos variables cualesquiera, nos queda un sistema de 3x3, el cual admite una solución si los vectores correspondientes son linealmente independientes. Dicha solución, conocida como "solución básica factible", corresponde a la base del espacio tridimensional formado por dichos vectores, y a su vez es un vértice del conjunto factible.

Si en nuestro problema hacemos y_4, y_5 iguales a cero, obtenemos como solución básica factible $y_1 = 4,\ y_2 = 2,\ y_3 = 1$, la cual concuerda con el "vértice" C del Gráfico, siendo $y_3 = 1$ la holgura de la restricción 1 que está inactiva, mientras que $y_4, y_5 = 0$ implican que las otras restricciones se cumplen con igualdad.

El paso siguiente es verificar si la solución obtenida es la óptima. Ello puede hacerse sustituyendo los valores de y_1, y_2 obtenidos a partir de (1.10)-(1.12) en la función objetivo (1.9):

$$\begin{aligned} z &= 2(4 + y_4 - y_5) + 3(2 - 2y_4 + y_5) \\ &= 14 - 4y_4 + y_5. \end{aligned} \qquad (1.15)$$

y viendo si *z* puede incrementarse aumentando una variable no básica. En nuestro caso tenemos que:

$$\frac{\partial z}{\partial y_4} = -4 < 0 \quad \text{y} \quad \frac{\partial z}{\partial y_5} = 1 > 0\,. \qquad (1.16)$$

lo cual nos indica que no estamos en una solución óptima, ya que *z* podría incrementarse aumentando y_5.

El próximo paso es entonces moverse a otro "vértice", para lo cual hay que cambiar de base o, lo que es lo mismo, incorporar una variable no básica reemplazando una básica. El criterio a seguir es el de incorporar a la base aquella variable no básica que pueda producir el mayor incremento marginal en la función objetivo. En nuestro caso, dicha variable es obviamente y_5, como se desprende de (1.16). La selección de la variable básica a ser reemplazada es un poco más demandante. La idea es eliminar aquella que restringe más el potencial incremento de *z* cuando se contempla la incorporación de la nueva variable básica y_5, recordando que la condición (1.13) exige que las variables sean todas positivas. Para ello expresamos las variables básicas en función de las no básicas y analizamos el sistema resultante:

$$y_1 = 4 + y_4 - y_5 \geq 0 \quad (1.17)$$

$$y_2 = 2 - 2y_4 + y_5 \geq 0 \quad (1.18)$$

$$y_3 = 1 - 1.5y_4 - 0.5y_5 \geq 0\,. \quad (1.19)$$

En primer lugar, de (1.16) sabemos que y_4 debe ser igual a cero ya que su incremento reduciría el valor de nuestra función objetivo. En consecuencia, vemos que la última desigualdad es la que más restringe el incremento potencial de y_5 ($y_5 \leq 2$), lo cual es indicativo de la conveniencia de sacar y_3 de la base. La nueva base estará conformada entonces por y_1, y_2, y_5. Resolviendo nuevamente el sistema de 3x3 resultante obtendremos como solución el punto B de nuestro grafico. Chequeando nuevamente la solución correspondiente, obtenemos:

$$\frac{\partial z}{\partial y_3}, \frac{\partial z}{\partial y_4} < 0 \quad (1.20)$$

lo cual indica que estamos en el óptimo, ya que no es posible mejorar *z* a través de la incorporación a la base de variables no básicas.

En términos generales, el método Simplex operará hasta que $\partial z / \partial y_j \leq 0$ para todas las variables no básicas. En el caso de que exista más de una variable no básica con $\partial z / \partial y_j > 0$, se deberá elegir alguna de ellas para incorporar a la base. Un criterio podría ser incorporar aquella variable que implique la derivada $\partial z / \partial y_j > 0$ más elevada. Sin embargo, esto no garantiza que dicha variable sea la que provoque el mayor incremento final en la función objetivo *z*, ya que el límite de cuánto pueda incrementarse la variable que entrará a la base viene dado, como vimos más arriba, por las restricciones que operan sobre las variables actualmente en la base. El caso de tener $\partial z / \partial y_j = 0$ para una variable no básica implica que la incorporación de ésta variable a la base no mejorará la función objetivo, e indica que existen múltiples óptimos entre el vértice actual y el que resultaría de incorporar y_j a la base. En términos del Gráfico 2 esto sucedía cuando las curvas de nivel de la función objetivo eran paralelas a alguna restricción.

Finalmente, si el problema a resolver es uno de minimización en vez de maximización, las variables de holgura deben incorporarse con signo negativo en vez de positivo, mientras que las desigualdades para la selección del óptimo deben revertirse.

1.2) Implementación de problemas de programación lineal en GAMS:[6]

El problema formado por las ecuaciones (1.3)-(1.7) de la sección anterior puede representarse en GAMS de la siguiente manera:

[6] Para una introducción a GAMS, ver Brooke et al. (1998).

```
1  Variables z, y1, y2;
   Equations objetivo, restri1, restri2, restri3, nonegat_y1, nonegat_y2;
   objetivo.. z =E= 2*y1 + 3*y2 ;
   restri1.. 0.5*y1 + y2 =L= 5 ;
   restri2.. y1 + y2 =L= 6 ;
   restri3.. 2*y1 + y2 =L= 10  ;
   nonegat_y1.. y1 =G= 0;
   nonegat_y2.. y2 =G= 0;
2  MODEL plineal /ALL/ ;
3  SOLVE plineal USING LP MAXIMIZING Z;
```

Si bien GAMS admite cierta flexibilidad a la hora de representar el problema, se suele adoptar el siguiente orden y clasificación para el ingreso de la información. En primer lugar (parte 1), declaramos y definimos los elementos que forman el problema, a saber: conjuntos, parámetros, variables, ecuaciones y su respectiva clasificación bajo las categorías "**Sets**", "**Parameters**", "**Variables**" y "**Equations**". Nótese que los elementos dentro de las categorías están separados por comas y que finalizada la enumeración se utiliza un punto y coma. En nuestro ejemplo no utilizamos "**Sets**", como podría ser el caso de modelos donde las variables tuvieran subíndices. Asimismo, los parámetros del modelo, representados por los coeficientes que acompañan las variables del problema, ya han sido valorizados directamente en el bloque de ecuaciones. En segundo lugar (parte 2), declaramos el nombre del modelo y especificamos las ecuaciones que lo conforman. En nuestro caso el modelo se llama "plineal" y esta formado por todas las ecuaciones antes mencionadas (/all/). Hubiera sido indistinto la enumeración de cada una de ellas entre barras y separadas por comas, o la enumeración de un subgrupo de ecuaciones, especificando entonces un submodelo a resolver. Finalmente (parte 3), con el comando "**SOLVE**" declaramos el tipo de problema a resolver ("LP" por "lineal programming") y la variable a maximizar.[7]

La sintaxis de las ecuaciones es bastante sencilla. Solo cabe mencionar el uso de =E= para designar una igualdad , =L= para el menor o igual y =G= para el mayor o igual. Nótese asimismo que cada ecuación está separada de otra por un punto y coma. Una forma alternativa que podría haberse utilizado para imponer las restricciones de no negatividad que operan sobre las variables y_1, y_2 es la siguiente:

```
y1.lo=0.0;
y2.lo=0.0;
```

donde el sufijo .lo indica el mínimo valor que puede adoptar la variable y_i, que en nuestro caso es igual a 0.

Luego de su ejecución GAMS genera dos archivos con información relevante sobre la resolución del problema: un archivo con extensión .log y otro con extensión .lst. El archivo .log contiene información sobre la ejecución del solver que por el momento no profundizaremos. Los resultados de la solución del modelo son expuestos en el archivo .lst.

[7] El solver que GAMS provee por defecto para resolver problemas de programación lineal es BDMLP, el cual es una implementación directa del algoritmo Simplex.

Este archivo repite inicialmente las ecuaciones del modelo y resume información del mismo. Finalmente arroja lo siguiente:

```
                  S O L V E      S U M M A R Y

     MODEL   plineal             OBJECTIVE  z
     TYPE     LP                      DIRECTION  MAXIMIZE
     SOLVER  BDMLP               FROM LINE  10

1    **** SOLVER STATUS     1 NORMAL COMPLETION
     **** MODEL STATUS      1 OPTIMAL
     **** OBJECTIVE VALUE               16.0000

      RESOURCE USAGE, LIMIT          0.000      1000.000
      ITERATION COUNT, LIMIT         3         10000

      BDMLP 1.3     Feb 14, 2001 WIN.BD.BD 19.8 056.043.039.WAT

      Originally developed by
       A. Brooke, A. Drud, and A. Meeraus,
       World Bank, Washington, D.C., U.S.A.

      Work space allocated           --    0.02 Mb

      EXIT -- OPTIMAL SOLUTION FOUND.

                            LOWER     LEVEL     UPPER    MARGINAL

2    ---- EQU objetivo        .         .         .        1.000
     ---- EQU restri1        -INF      5.000     5.000     2.000
     ---- EQU restri2        -INF      6.000     6.000     1.000
     ---- EQU restri3        -INF      8.000    10.000      .
     ---- EQU nonegat_y1      .        2.000     +INF       .
     ---- EQU nonegat_y2      .        4.000     +INF       .

                            LOWER     LEVEL     UPPER    MARGINAL

     ---- VAR z              -INF     16.000     +INF       .
     ---- VAR y1             -INF      2.000     +INF       .
     ---- VAR y2             -INF      4.000     +INF       .


     **** REPORT SUMMARY :        0     NONOPT
                                  0 INFEASIBLE
                                  0  UNBOUNDED


     EXECUTION TIME       =        0.050 SECONDS    0.7 Mb
     WIN198-120
```

Mucha de la información que se obtiene resulta de fácil interpretación. Sin embargo, vale la pena detenerse en algunos puntos. Inicialmente (parte 1) GAMS nos informa que la resolución se completó sin complicaciones `(SOLVER STATUS  1 NORMAL COMPLETION)`, que se ha encontrado un punto óptimo `(MODEL STATUS  1 OPTIMAL)` y que la función objetivo adopta en dicho punto el valor 16 `(OBJECTIVE VALUE  16.0000)`.

Asimismo (parte 2) GAMS proporciona información acerca de las ecuaciones y variables que conforman el problema. Cada ecuación y cada variable tienen asociados 3 valores: mínimo `(LOWER)`, máximo `(UPPER)`, "nivel" `(LEVEL)` y marginal `(MARGINAL)`. Los valores mínimos y máximos reflejan las restricciones que pesan sobre las ecuaciones o variables. En nuestro caso vemos que el valor máximo que puede adoptar la restricción número uno es 5 y que además no está restringida inferiormente, es decir que la cota inferior es "menos infinito" `(-INF)`. El valor 0 está representado por un punto. Puede llamar la atención el hecho de que las variables y_1 y y_2 no estén acotadas inferiormente (`-INF`); sin embargo, se debe recordar que están restringidas en forma indirecta a través de las ecuaciones `nonegat_y1 y nonegat_y2` cuyo valor mínimo es 0. En el caso de utilizar el sufijo .lo, como fuera mencionado anteriormente, las variables y_i tendrían asociados valores iguales a 0.

El "nivel" representa el valor que adoptó la variable o ecuación en el punto que constituye el óptimo del problema. En nuestro caso es $y_1 = 2$, $y_2 = 4$ y $z = 16$, que concuerda con el punto B del Gráfico 2. Por su parte las dos primeras restricciones adoptan un "nivel" igual al valor máximo que pueden alcanzar, es decir que las mismas son operativas en ese punto, por lo que las variables de holgura asociadas (y_3, y_4) son iguales a 0. La restricción 3, en cambio, no es operativa y su variable de holgura adopta el valor 2.

Finalmente, el valor marginal nos dice cuánto aumenta el valor de z si relajamos en una unidad las restricciones o las variables de holgura. Como es de esperar, aquellas restricciones que no son operativas (como la restricción 3) tendrán un valor marginal igual a 0. El valor marginal de las restricciones refleja indirectamente qué esta sucediendo con las variables de holgura, ya que relajar en una unidad una restricción equivale a reducir en una unidad una variable de holgura. Por ejemplo, el valor marginal 2 correspondiente a la restricción 1 se puede pensar alternativamente como:

$$\frac{\partial z}{\partial y_3} = -2. \tag{1.21}$$

De la misma manera, para la restricción 2 tenemos:

$$\frac{\partial z}{\partial y_4} = -1. \tag{1.22}$$

Estas dos condiciones eran de esperar ya que es necesario que ambas variables no básicas posean derivadas parciales negativas en el óptimo.

Los valores asociados a cada variable y ecuación son esenciales a la hora de identificar las soluciones del problema, en particular si es que existen múltiples soluciones. Supongamos que en nuestro ejemplo la función objetivo pasa a ser $z = y_1 + y_2$, de tal manera que las curvas de nivel posean pendiente –1 al igual que la restricción 2. En éste

caso habría infinitas soluciones representadas por el segmento BC del Grafico 1. GAMS nos daría los siguientes resultados:

```
EXIT -- OPTIMAL SOLUTION FOUND.

                        LOWER     LEVEL     UPPER    MARGINAL

---- EQU objetivo         .         .         .        1.000
---- EQU restri1        -INF      4.000     5.000       .
---- EQU restri2        -INF      6.000     6.000     1.000
---- EQU restri3        -INF     10.000    10.000      EPS
---- EQU nonegat_y1      .        4.000     +INF        .
---- EQU nonegat_y2      .        2.000     +INF        .

                        LOWER     LEVEL     UPPER    MARGINAL

---- VAR z              -INF      6.000     +INF        .
---- VAR y1             -INF      4.000     +INF        .
---- VAR y2             -INF      2.000     +INF        .
```

El solver BDMLP aplica el algoritmo Simplex hasta que encuentra una solución. En éste caso se detiene en el punto y_1 =4, y_2 =2 y z = 6 que concuerda con el punto C del grafico 2. GAMS no nos informa que existen otros puntos que constituyen una solución del problema.

Sin embargo, nosotros podemos deducir eso a partir de la información asociada a cada ecuación. En primer lugar, a partir de los valores "LEVEL" y "UPPER" asociados a cada ecuación, sabemos que las restricciones activas son la segunda y la tercera. Asimismo, el "MARGINAL" de la tercera ecuación posee la sigla "EPS" que significa que el valor es cercano a cero, lo cual nos indica que la variable de holgura asociada a esta restricción (y_5) que no forma parte de la base podría ingresar a la misma sin afectar el valor de z, obteniéndose un nuevo punto optimo (el vértice B del Gráfico 2). De hecho, si nosotros imponemos restricciones adicionales sobre el problema de tal manera de obligar al solver a detenerse en el punto B (imponiendo por ejemplo las condiciones y1.lo = 2 y y2.lo = 4) observaríamos que el valor de z se mantendría constante. Lo mismo ocurriría en cualquier punto del segmento BC.

2.1) Programación No Lineal[8]

La programación no lineal trata con problemas de optimización de una función no lineal sujeta a desigualdades/igualdades lineales o no lineales. En forma general el problema puede formularse como el de maximizar:

$$B = f(x_1, x_2, ..., x_n) \tag{2.1}$$

sujeto a:

$$\begin{aligned} & g^1(x_1, x_2, ..., x_n) \le c_1 \\ & g^2(x_1, x_2, ..., x_n) \le c_2 \\ & ..... \\ & g^m(x_1, x_2, ..., x_n) \le c_m. \end{aligned} \tag{2.2}$$

Adicionalmente, en aplicaciones económicas se suele agregar la condición de no negatividad de las variables:

$$x_i \ge 0 \quad (\text{i=1, 2,..., n}). \tag{2.3}$$

Tanto las funciones *f* como g^j se suponen continuas y derivables.[9]

Existen condiciones como las de Kuhn-Tucker que permiten caracterizar cualitativamente la solución del problema en términos de su existencia o no. [10] Sin embargo, las mismas no son fáciles de chequear para problemas de cierta complejidad. Esto

---

[8] Para un tratamiento extensivo ver, Baazara et al. (1995), Chiang (1987), Judd (1998), Rardin (1998) y Silberberg (2001).

[9] GAMS provee solvers específicos para trabajar con problemas de programación entera cuyo tratamiento excede esta sección. Ver GAMS -The Solver Manuals (2001)

[10] Definiendo la función Lagrangeana Z como:

$$Z = f(x_1, ..., x_n) + \sum_{i=1}^{m} y_i \left[r_i - g^i(x_1, ..., x_n) \right]$$

donde y_i son los correspondientes multiplicadores de Lagrange, las condiciones de Kuhn-Tucker para la maximización de la función objetivo son

$$\partial Z/\partial x_j \ge 0, \ \partial Z/\partial y_i \ge 0, \ x_j \ge 0, \ y_i \ge 0,$$

$$x_j \, \partial Z/\partial x_j = 0, \ y_i \, \partial Z/\partial y_i = 0. \ (i = 1, ...m \ ; \ \text{j=1,...n}).$$

Para el caso de minimización, se revierten las desigualdades $\partial Z/\partial x_j$ y $\partial Z/\partial y_i$.

Cuando las restricciones del problema son lineales o cuando cumplen ciertos requisitos especiales, estas condiciones son necesarias para un óptimo. Cuando en el ortante no negativo la función objetivo es diferenciable y cóncava [convexa] mientras que cada restricción es diferenciable y convexa [cóncava], tales condiciones son suficientes para un máximo [mínimo]. Finalmente cuando todas estas características del problema están presentes las condiciones de Kuhn-Tucker son necesarias y suficientes. Ver Chiang (1987), Simon y Blume (1994).

hace que, en general, los problemas de programación no-lineal requieran para su solución de métodos computacionales.

Dada la variedad de problemas que engloba la programación no lineal existen varios algoritmos de solución de acuerdo a las características de cada clase de problemas. Mas allá de ésta diversidad, los solvers de GAMS generalmente utilizan algún refinamiento y/o combinación de tres algoritmos básicos: el método gradiente, el método de Newton y el método de "penalidades". Introduciremos los dos primeros métodos en un contexto de optimización no restringida para luego ver como cambian al imponer restricciones. El método de "penalidades" es propio de problemas de optimización restringida y será tratado al final de la sección.

Tanto el método gradiente como el de Newton se basan en métodos numéricos de búsqueda que, partiendo de un punto y mediante el ensayo repetido y sistemático de distintos valores de las variables de elección del problema, buscan nuevos puntos que impliquen una mejora en la función objetivo. Dicha mejora se define como un incremento en una maximización o un decremento en una minimización. En otras palabras, ambos métodos pueden pensarse como procedimientos de búsqueda de "direcciones" hacia donde tienen que moverse la variables de elección.

Los dos métodos siguen un esquema básico común. Se parte de un vector solución inicial de variables de elección x^0, y a partir de allí se pasa a un nuevo vector solución $x^1 = x^0 + \lambda \Delta x$, donde Δx representa la dirección del cambio y λ el tamaño (positivo) del movimiento. Δx debe ser una dirección que mejore (localmente) la función objetivo y λ será elegido respetando las restricciones del problema pero procurando maximizar el incremento generado por Δx. Una vez hallado, x^1 será nuestro nuevo vector solución inicial y a partir de allí el algoritmo buscará una nueva dirección de cambio. En el caso de que ninguna dirección produzca una mejora significativa en la función objetivo el algoritmo se detendrá y tomará a éste vector como un posible óptimo local (o global si el problema cumple ciertas condiciones de concavidad o convexidad, según corresponda).

Estos métodos modificarán Δx y λ en base a la experiencia de puntos ya visitados y a la información local sobre la forma de la función objetivo en un entorno del punto actual. Esta información local es obtenida mediante las derivadas parciales sucesivas de la función objetivo en el punto en cuestión: recordemos, por ejemplo, que las derivadas primeras o gradientes en un punto nos proporcionan información acerca de cómo cambia la función objetivo en el entorno de un punto. Asimismo, las segundas derivadas o Hessiano nos proporcionan información acerca del cambio en la pendiente o curvatura de la función en el entorno de un punto. Esta información local se ve resumida en la aproximación de Taylor, nos permite conocer el impacto en la función objetivo de un cambio infinitesimal en las variables independientes en el entorno de x. Despreciando los términos de orden mayor que dos que tienden rápidamente a 0, la aproximación de Taylor viene dada por:

$$f(\underbrace{x^0 + \lambda \Delta x}_{x^1}) \cong f(x^0) + \lambda \nabla f(x^0) \Delta x + \frac{\lambda^2}{2} \Delta x^T H(x^0) \Delta x . \qquad (2.4)$$

donde $\nabla f(x^0)$ y $H(x^0)$ representan, respectivamente, el vector gradiente y la matriz Hessiana evaluados en x^0, y donde Δx^T representa el vector traspuesto de las diferencias.

2.2) El método gradiente.

Recapitulando, la cuestión esencial es encontrar una dirección de cambio que implique una mejora (local) en $f(x)$. Tanto el método gradiente como el de Newton toman la aproximación de Taylor (2.4) como punto de partida para ésta búsqueda. El método gradiente, sin embargo, sólo toma hasta el término de primer orden de la expresión, es decir:

$$f(x^0 + \lambda\Delta x) \cong f(x^0) + \lambda\nabla f(x^0)\Delta x\,. \tag{2.5}$$

Resulta evidente de (2.5) que una dirección Δx que mejoraría la función objetivo, para un tamaño λ positivo, sería $\Delta x = \nabla f(x^0)$ en el caso de un problema de maximización y $\Delta x = -\nabla f(x^0)$ en uno de minimización. De esta manera obtendríamos:

$$f(x^0 + \lambda\Delta x) \cong f(x^0) + \lambda(\nabla f(x^0))^2 > f(x^0)\,. \tag{2.6}$$

que implica una mejora en el primer caso y:

$$f(x^0 + \lambda\Delta x) \cong f(x^0) - \lambda(\nabla f(x^0))^2 < f(x^0)\,. \tag{2.7}$$

que implica una mejora en el segundo caso. En el caso de que $\nabla f(x^0) \cong 0$, podremos decir que encontramos un punto crítico, posible candidato a ser un óptimo local.

En síntesis, el método gradiente toma al vector gradiente como dirección base de su movimiento. Esta dirección a su vez tiene la propiedad de ser la que produce localmente la tasa de mejora más alta en la función objetivo.[11]

Sin embargo, es importante destacar que en muchos casos, a medida que se acerca al punto critico, el método pierde efectividad ya que comienza a zigzaguear y se aproxima muy lentamente. Esto ultimo se debe a que al acercarnos al punto crítico la forma de la función objetivo suele cambiar muy rápidamente, lo cual hace que en poca distancia el algoritmo cambie de dirección en forma dramática.[12] Esto se ilustra en el Gráfico 3, donde

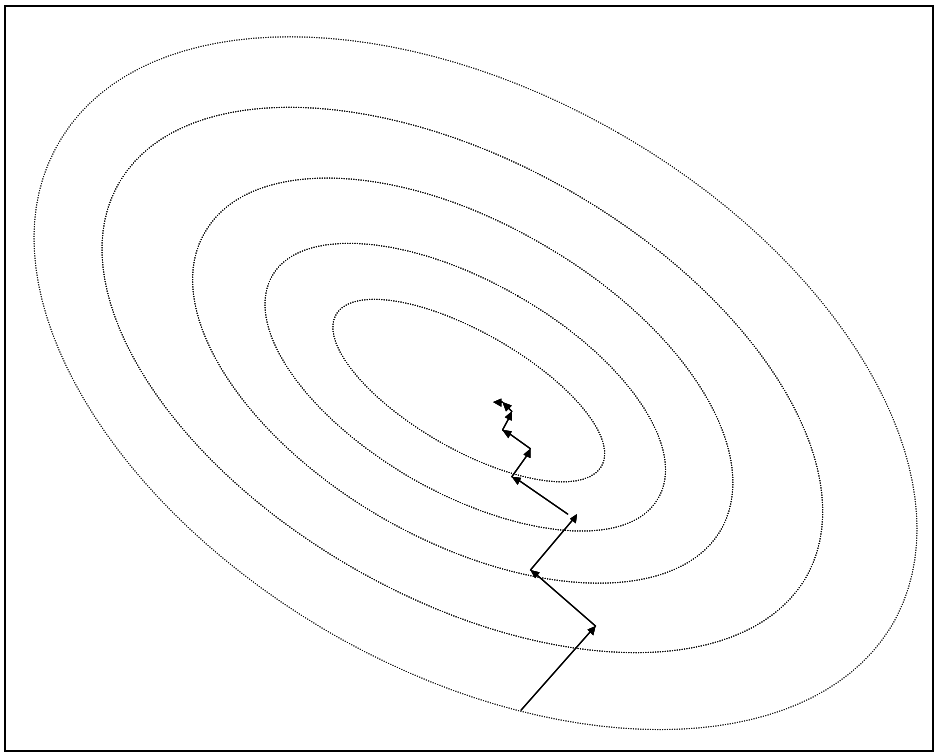
[11] Esto se debe a que los vectores gradientes resultan perpendiculares a las curvas de nivel de la función objetivo

[12] Siguiendo a Rardin (1998) podemos sintetizar esquemáticamente los pasos que sigue el algoritmo en el caso de optimización libre:

Paso 0: Elección de un vector solución inicial x^0 y un margen de tolerancia $\varepsilon > 0$ dentro del cual el algoritmo detiene sus iteraciones

se representan las curvas de nivel de una función de dos variables independientes y la secuencia de aproximación de vectores gradiente al punto crítico.

Gráfico 3



2.3 El método de Newton.

Si queremos evitar el lento zigzaguear propio del método gradiente debemos incorporar más información sobre el entorno del punto en cuestión. A tal fin el método de Newton utiliza una aproximación de Taylor de segundo orden de la función objetivo como en (2.4). A diferencia de la expresión (2.5), que es lineal en términos de Δx, (2.4) es cuadrática y por lo tanto admite la posibilidad de tener un optimo local. De esta manera podemos determinar el movimiento $\lambda \Delta x$ como el máximo (o mínimo) de la aproximación

---

Paso 1: Cálculo del vector gradiente de la función objetivo [$\nabla f(x^0)$]evaluado en el punto inicial x^0

Paso 2: si la norma del vector gradiente $\left\|\nabla f(x^0)\right\| < \varepsilon$, detener la ejecución del algoritmo. El punto x^0 esta lo suficientemente cercano a un punto crítico.

Paso 3: tomar al vector gradiente como dirección base de movimiento: $\Delta x = \pm \nabla f(x^0)$ con (+) para maximización y (-) para minimización

Paso 4: para elegir el tamaño optimo de movimiento λ, debemos resolver el siguiente problema $\max / \min\ f(x^0 + \lambda \Delta x)$,

Paso 5: calcular el nuevo vector solución $x^1 \rightarrow x^0 + \lambda \Delta x$

Paso 6: volver al paso numero 1 tomando a x^1 como nuevo punto inicial

de Taylor. Fijando $\lambda = 1$, diferenciado la expresión (2.4) con respecto a Δx e igualando a 0 obtenemos:

$$\nabla f(x^0) + H(x^0)\Delta x = 0 \tag{2.8}$$

o lo que es lo mismo:

$$\Delta x = [-H(x^0)]^{-1}[\nabla f(x^0)] \tag{2.9}$$

que será el movimiento óptimo utilizado en el método de Newton.

Este método garantiza una convergencia al punto crítico con menos iteraciones que el método gradiente[13]. Sin embargo, queda claro que computar la matriz Hessiana en cada iteración y luego invertirla para calcular Δx implica un importante esfuerzo computacional. Además, el método de Newton sólo converge al punto óptimo si empieza a iterar relativamente cerca de dicho punto: la dirección de Newton no es necesariamente una dirección de ascenso o descenso como la dirección del gradiente. A veces la información adicional que provee la aproximación de Taylor de segundo orden es muy pobre y el método puede sugerir direcciones de movimiento que no necesariamente mejoren la función objetivo. Cuando la dirección de Newton presenta éstas dificultades se la suele reemplazar por la dirección del gradiente. Asimismo, pueden surgir problemas a la hora de computar las matriz Hessiana (esta puede ser singular) y al resolver el sistema de ecuaciones para hallar Δx.

Algunos de éstos problemas pueden aminorarse utilizando métodos llamados de cuasi-Newton, que sustituyen el Hessiano por otras matrices cuyo cálculo es mas sencillo y que reflejan en alguna medida las propiedades esenciales del Hessiano.[14] Esta matriz se construirá utilizando información sobre las primeras derivadas en los puntos sucesivos y deberá cumplir con ciertas condiciones. Sabiendo que el Hessiano refleja la tasa de cambio de las primeras derivadas, o sea:

$$\nabla f(x^k) - \nabla f(x^{k-1}) \cong H(x^k)(x^k - x^{k-1}). \tag{2.10}$$

Entonces:

$$H(x^0)^{-1}[\nabla f(x^1) - \nabla f(x^1)] \cong (x^1 - x^0) \cong \Delta x. \tag{2.11}$$

[13] La mayor velocidad de convergencia de este método se evidencia particularmente cuando se optimiza una función cuadrática, ya que Newton llega al óptimo en tan solo una iteración

[14] Todos estos métodos forman parte de lo que se conoce como "Algoritmos de Métrica Variable" que en general proponen como dirección de movimiento la siguiente: $\Delta x = -D\nabla f(x^0)$, donde D es una matriz de "desviación". En el caso de método gradiente D es la matriz identidad y con Newton es el Hessiano.

por lo tanto la matriz Hessiana aproximada H_k deberá satisfacer la condición (2.11). Asimismo, deberá preservar la propiedad de ser simétrica y, para garantizar que la dirección mejore la función objetivo, deberá ser definida negativa en caso de maximización y definida positiva en caso de minimización.[15]

Existen varios esquemas para construir matrices $\tilde{H}_k$ que cumplan con los requerimientos anteriormente mencionados. Uno de los más difundidos es el de Broyden-Fletcher-Golfarb-Shanno (BFGS) que utiliza el siguiente mecanismo iterativo:

$$\tilde{H}_{k+1} = \tilde{H}_k - \frac{\tilde{H}_k z_k z_k^T \tilde{H}_k}{z_k^T \tilde{H}_k z_k} + \frac{y_k y_k^T}{y_k^T z_k} \qquad (2.12)$$

donde $z_k = (x^{k+1} - x^k)$ e $y_k = (\nabla f(x^{k+1}))^T - (\nabla f(x^k))^T$. Dado que la secuencia de matrices $\tilde{H}_k$ puede no converger al verdadero valor de H_k, eventualmente resulta necesario computar el verdadero Hessiano al final de las iteraciones.[16]

2.3) Modificaciones para el caso de optimización restringida.

2.3.1) Método Gradiente Reducido

Tanto el método de gradiente como el de Newton parten de un punto inicial e iteran buscando nuevos puntos que mejoren la función objetivo. A diferencia del caso de optimización libre presentado anteriormente, en el caso de optimización restringida la dirección de búsqueda que adopta el algoritmo no solo deberá mejorar la función objetivo sino que además deberá ser factible, es decir, deberá respetar las restricciones impuestas al problema.

---

[15] De esa manera la dirección de movimiento $\nabla f(\Delta x^k)\Delta x$ sería $\nabla f(\Delta x^k)(-H_k^{-1})\nabla f(\Delta x^k)^T > 0$ en el caso de maximización y $\nabla f(\Delta x^k)(-H_k^{-1})\nabla f(\Delta x^k)^T < 0$ en el de minimización

[16] Siguiendo a Judd (1999) y Rardin (1998) podemos sintetizar la operatoria del algoritmo de cuasi-Newton-BFGS de la siguiente manera:

Paso 0: Elección de un vector solución inicial x^0, un hessiano inicial (u otra matriz inicial como por ejemplo la matriz identidad) y un margen de tolerancia $\varepsilon > 0$ dentro del cual el algoritmo detiene sus iteraciones

Paso 1: Cálculo del vector gradiente de la función objetivo [$\nabla f(x^0)$] evaluado en el punto inicial x^0

Paso 2: si la norma del vector gradiente $\left\|\nabla f(x^0)\right\| < \varepsilon$, detener la ejecución del algoritmo. El punto x^0 esta lo suficientemente cercano a un punto estacionario.

Paso 3: tomar como dirección base de movimiento: $\Delta x = [-H(x^0)]^{-1}[\nabla f(x^0)]$

Paso 4: calcular el nuevo vector solución $x^1 \rightarrow x^0 + \lambda \Delta x$

Paso 5: recalcular el Hessiano aproximado utilizando la regla (2.12).

Paso 6: volver al paso numero 1 tomando a x^1 como nuevo punto inicial

El efecto de las restricciones puede ilustrarse con el método gradiente reducido, aplicado a optimizar una función no lineal derivable sujeta a restricciones de igualdad lineales. Es decir maximizar [o minimizar]:

$$f(X) \tag{2.13}$$

sujeto a:

$$\begin{aligned} &\text{Ax=b} \\ &\text{x} \geq 0. \end{aligned} \tag{2.14}$$

Recordemos que si poseemos restricciones de desigualdad éstas pueden ser transformadas, como vimos en el caso de la programación lineal, en restricciones de igualdad utilizando variables de holgura. La condición adicional que ahora deberá respetar la dirección Δx elegida en cada iteración es, para el caso de restricciones lineales:

$$A\Delta x = 0 \tag{2.15}$$

donde $\Delta x_j \geq 0$ para todo j con $x_j = 0$, es decir que el cambio no implique violar la igualdad $Ax=b$, ni la restricción de no negatividad de los elementos del vector x.

Al igual que el método Simplex, este método distingue inicialmente entre variables básicas y no básicas, es decir que separa el sistema de restricciones lineales en componentes dependientes e independientes. Si embargo, a diferencia del método Simplex, podemos tener variables no básicas con valores positivos, pues el óptimo no necesariamente se encuentra en un vértice de la región factible ya que la función objetivo puede alcanzar un óptimo local antes de alcanzar la "frontera". Estas variables no básicas se denominan superbásicas y en general no superan en número a la cantidad de variables no lineales. Entonces, podemos descomponer la condición (2.15) de la siguiente manera:

$$A\Delta x = B\Delta x^{(B)} + S\Delta x^{(S)} + N\Delta x^{(N)} = 0 \tag{2.16}$$

donde B, S y N son las submatrices de A formadas por las columnas correspondientes a variables básicas, superbásicas y no básicas.

El método de gradiente reducido considera inicialmente a las variables superbásicas x_s como variables "libres" que pueden moverse en la dirección deseada de tal manera de mejorar la función objetivo. Las variables básicas se ajustarán para mantener la condición $A\Delta x = 0$, por lo tanto se considera que $\Delta x^{(N)}$ es 0. Por lo dicho anteriormente podemos reexpresar (2.16) de la siguiente manera:

$$\Delta x^{(B)} = -B^{-1}S\Delta x^{(S)} \tag{2.17}$$

lo cual nos permite tener una expresión de la dirección Δx que incluye la condición de factibilidad $A\Delta x = 0$:

$$\Delta x = \Delta x^{(B)} + \Delta x^{(S)} + \Delta x^{(N)} = (-B^{-1}S \quad I \quad 0)\Delta x^{(S)} = Z\Delta x^{(S)} \tag{2.18}$$

donde Z es el vector $(-B^{-1}S \quad I \quad 0)$.

En el caso de optimización no restringida el método gradiente sugería (ver sección 2.1) que la dirección Δx a seguir para optimizar la función objetivo debía ser $\Delta x = \nabla f(x^0)$ en el caso de un problema de maximización y $\Delta x = -\nabla f(x^0)$ en uno de minimización. En éste caso la dirección sugerida es:

$$\Delta x^{(S)} = \pm Z^T \nabla f(x) \tag{2.19}$$

dependiendo de si el problema es de maximización (+) o minimización (-).[17]

Si no se puede mejorar la función objetivo con la actual configuración de variables básicas, no básicas y superbásicas, algunas variables $x^{(N)}$ son transformadas en $x^{(S)}$ de acuerdo a su aporte a la función objetivo, y el proceso de obtención del vector gradiente reducido se repite, moviendo las variables superbásicas y ajustando las básicas. Si en el proceso alguna de las variables $x^{(B)}$ llega a adoptar el valor 0, automáticamente son transformadas en variables $x^{(N)}$ y una variable, preferentemente superbásica, toma su lugar en la base.

Una vez obtenida la dirección del cambio como figura en (2.19), debemos obtener el tamaño óptimo de movimiento λ a partir de la resolución del problema de maximizar o minimizar:

$$f(x^0 + \lambda \Delta x) \tag{2.20}$$

sujeto a:

$$0 \le \lambda \le \lambda_{\max} \tag{2.21}$$

donde $\lambda_{\max}$ es el máximo valor de λ tal que $x_j \ge 0$. El proceso continuará hasta que $|\Delta x| \le \varepsilon$, donde ε es un valor o "límite de tolerancia" preestablecido. En este caso se estará

---

[17] En efecto, si linealizamos por Taylor y sustituimos por la expresión (2.18) obtenemos:

$$f(x^0 + \lambda \Delta x) \cong f(x^0) + \lambda \nabla f(x^0) \Delta x = f(x^0) + \lambda \nabla f(x^0) Z \Delta x^{(S)}.$$

Por lo tanto, si adoptáramos la dirección sugerida en (2.19) obtendríamos las siguiente expresión:

$$f(x^0 + \lambda \Delta x) \cong f(x^0) + \lambda \underbrace{\nabla f(x^0) Z Z^T \nabla f(x^0)}_{\ge 0} > f(x^0)$$

para la maximización de f y

$$f(x^0 + \lambda \Delta x) \cong f(x^0) - \lambda \underbrace{\nabla f(x^0) Z Z^T \nabla f(x^0)}_{\ge 0} < f(x^0)$$

para la minimización de *f*.

lo suficientemente cerca de un punto estacionario. Por último, vale la pena mencionar que este método adolece de las mismas dificultades que su versión para problemas no restringidos, a saber, su lenta convergencia. Esta dificultad puede aminorarse utilizando métodos de Newton o cuasi- newton. Reexpresando la aproximación de Taylor (1.16) para $\lambda = 1$ obtenemos:

$$\begin{aligned} f(x^0 + \Delta x) &\cong f(x^0) + \nabla f(x^0)\Delta x + \Delta x^T H(x^0)\Delta x \\ &= f(x^0) + \nabla f(x^0)(Z\Delta x^{(S)}) + (Z\Delta x^{(S)})^T H(x^0)(Z\Delta x^{(S)}) \end{aligned} \tag{2.22}$$

y derivando esa expresión con respecto a $\Delta x^{(S)}$ e igualando a 0 obtenemos:

$$\nabla f(x^0)Z + ZH(x^0)Z\Delta x^{(S)} = 0 \tag{2.23}$$

Por lo tanto, la dirección $\Delta x^{(S)}$ óptima pasa a ser:

$$\Delta x^{(S)} = [ZH(x^0)Z]^{-1}[-\nabla f(x^0)Z] \tag{2.24}$$

2.3.2) Método de Penalidades

La idea básica de éste método es permitir que cualquier dirección Δx sea factible. Sin embargo, se modifica la función objetivo mediante una función de penalidad de tal manera de que resulte muy "costoso" violar las restricciones del problema. En síntesis, si tenemos el problema de maximizar o minimizar:

$$f(X) \tag{2.25}$$

sujeto a:

$$g(X) \underset{>}{\overset{<}{-}} b \tag{2.26}$$

lo transformamos en minimizar o maximizar:

$$\mathrm{F(X)} = f(X) \pm \delta \sum_i p_i(X) \tag{2.27}$$

donde utilizamos "+" para problemas de minimización y "–" para los de maximización, donde δ es un multiplicador de penalidad (positivo) y donde p_i son funciones de penalidad que adoptan valores positivos o nulos de acuerdo a si satisfacen o no la restricción *i*. Resulta evidente que la representación del problema (2.27) no es equivalente a (2.25) salvo en el límite de δ tendiendo a infinito. Sin embargo, en términos prácticos una buena aproximación al óptimo se obtiene para valores grandes de δ.[18]

Las funciones de penalidades pueden adoptar distintas formas. Resulta particularmente útil que éstas sean diferenciables de tal manera que la nueva función (no

[18] Se puede demostrar que, bajo ciertas condiciones, el óptimo del problema con penalización es igual al óptimo del problema original para valores finitos del multiplicador δ. Ver Armitano et al (1985) y Judd (1999).

restringida) del problema (2.27) pueda ser optimizada con los métodos ya vistos (gradiente – Newton). Asimismo, resulta imprescindible que la función cumpla con la condición de que si *X* pertenece a la región factible, entonces *F(X)=f(X)* y, en caso de no pertenecer, que *F(X)>>f(X),* donde >> indica "elemento a elemento". Por último, el mínimo de *F(X)* deberá ser el mínimo de *f(X).* Entre las formulaciones más comunes para la función de penalidad p_i podemos encontrar:

$$(\max[0, b_i - g_i(X)])^2 \text{ para restricciones de } \geq,$$
$$(\max[0, g_i(X) - b_i])^2 \text{ para restricciones de } \leq,$$
$$(g_i(X) - b_i)^2 \text{ para restricciones de igualdad.}$$

La operatoria iterativa del mecanismo de penalidades comienza por asignar inicialmente un valor de δ relativamente pequeño (baja penalidad) y un punto inicial x_0 donde comenzar la iteración. Dado el valor δ, se computan sucesivas direcciones de cambio óptimas Δx (utilizando por ejemplo el método gradiente) para resolver el problema (2.27). En el caso de que la sucesivas direcciones de movimiento Δx adoptadas violen en forma creciente las restricciones del problema, se aumentará gradualmente el valor de δ buscando mantener la factibilidad de la solución.[19] Usualmente, dado el valor pequeño y arbitrario de δ, es de esperar que inicialmente se violen las restricciones. La penalidad creciente de δ genera que cada Δx se ajuste cada vez mejor a las restricciones.

La utilidad computacional del planteo (2.27) frente a los planteos usuales de Lagrange o Kuhn-Tucker se aprecia particularmente cuando existen modelos grandes, (con muchas variables y restricciones) con restricciones de desigualdad. En este caso la resolución del problema con Kuhn-Tucker implicaría una costosa resolución de subsistemas de ecuaciones no lineales con sucesivas y diferentes combinaciones de restricciones activas y no activas buscando entre las soluciones factibles aquellas que optimicen *f(x).* Con el método de penalidades, en cambio, se debe resolver una sucesión de problemas no lineales no restringidos para valores crecientes de δ.

2.4) Programación no lineal en Gams.

Como fue planteado anteriormente, los solvers utilizados por GAMS aplican alguna combinación o refinamiento de los métodos anteriormente citados. A fin de ilustrar como opera un solver en particular, se utilizará el Minos5. [20] Este solver resuelve cada clase de problemas de optimización con un algoritmo distinto, a saber:

a) para problemas con funciones objetivo lineales y restricciones lineales aplica el método Simplex

[19] La lógica detrás del aumento gradual de δ es evitar que inicialmente la función sea demasiado inclinada y por tanto pequeños movimientos de x tengan un alto impacto en F ya que esto dificulta los cálculos , en particular el computo del Hessiano.
[20] Ver GAMS-The Solvers Manuals (2001).

b) para problemas con funciones objetivo no lineales y restricciones lineales aplica el método gradiente reducido combinado con métodos cuasi-Newton en el mismo estilo de la sección 2.3.1. Vale la pena mencionar que el caso de optimización libre está enmarcado dentro de éste algoritmo.

c) para problemas con funciones objetivo no lineales y restricciones no lineales utiliza un método gradiente reducido combinado con un método de penalidades. Se linealizan las restricciones no lineales (quedando un problema equivalente al del punto 2.3.2) y se modifica la función objetivo poniendo una función de penalidad cuadrática para corregir la diferencia entre la restricción linealizada y la restricción original. De ésta manera, en la secuencia del algoritmo existirán iteraciones "mayores" que implican la linealización de las restricciones, la modificación de la función objetivo con la función de penalidad y la resolución del subproblema planteado. Por cada iteración "mayor" existen iteraciones "menores", que están relacionadas con las sucesivas direcciones del método gradiente reducido buscando una solución al subproblema. El parámetro de penalidad δ es modificado en cada iteración "mayor" de tal forma de ir garantizando sucesivamente el cumplimiento de las restricciones.

A fin de ilustrar cómo opera el solver Minos5 con problemas no lineales, se analizará un mismo problema presentado en dos formas distintas. El problema es la maximización de beneficios de un monopolista que enfrenta un función de demanda lineal y una función de costo mínimo cuadrática.

1) Maximizar:

$$\pi = pq - c^* \tag{2.28}$$

sujeto a:

$$\begin{aligned} p &= 10 - 2q \\ c^* &= q^2 + 1. \end{aligned} \tag{2.29}$$

2) Maximizar:

$$\pi = pq - (q^2 + 1) \tag{2.30}$$

sujeto a:

$$p = 10 - 2q. \tag{2.31}$$

Minos5 resuelve el segundo caso (una función objetivo no lineal con restricción lineal) aplicando el algoritmo B, y el primer caso (función objetivo no lineal con restricción no lineal) con el algoritmo C.

2.4.1) Optimización de función no lineal sujeta a restricciones lineales:

En el siguiente cuadro podemos ver la representación del problema 2 en GAMS. No existen diferencias significativas respecto a la formulaciones lineales de la sección 1. Debe notarse solamente la forma de expresar q^2, como q**2. Nótese además que el problema se clasifica como uno de Non Lineal Programing (NLP).

```
Variables
Phi
Q
P
Equations
Objetivo
Demanda;
Objetivo.. phi =E= p*q - ((q**2)+1)  ;
Demanda.. p=E= 10-2*q;
MODEL non2 /ALL/ ;
SOLVE non2 using NLP maximizing phi;
```

Al igual que en el caso lineal, en la resolución del problema GAMS genera dos archivos: un archivo .lst con los resultados y un archivo de ejecución .log. Antes de pasar al análisis de los resultados, es interesante analizar el archivo .log:

```
MINOS5        Feb 14, 2001 WIN.M5.M5 19.8 108.043.039.WAT

 Work space allocated           --    0.04 Mb
 Reading data...
 Reading nonlinear code...

 Itn     0 -- Feasible solution.  Objective =  -2.600000000E+01


   Itn  Nopt  Ninf  Sinf,Objective  Nobj  NSB
     1     1     0  7.33333333E+00     6    1

 EXIT -- OPTIMAL SOLUTION FOUND

 Major, Minor itns              1           1
 Objective function       7.3333333333333E+00
 FUNOBJ, FUNCON calls           6           0
 Superbasics, Norm RG           1    1.78E-15
 Degenerate steps               0        0.00
 Norm X,      Norm PI    6.89E+00    2.37E+00
 Norm X,      Norm PI    8.20E+00    1.89E+00  (unscaled)
--- Restarting execution
--- NONLINEAL2.GMS(11) 0 Mb
--- Reading solution for model non2
*** Status: Normal completion
```

GAMS nos informa, entre otras cosas. que:

a) se ha encontrado una solución óptima `(OPTIMAL SOLUTION FOUND)`

b) el solver necesitó una iteración para resolver el problema (`Itn` 1)

c) existe una *no optimalidad*, esto significa que existe alguna variable que posee un marginal con valor distinto del optimo [21] (`Nopt 1`).

d) no se están violando las restricciones (`Ninf 0`) y el valor de la función objetivo en el optimo es `Objective 7.33333333E+00`.

e) Minos5 calculó el vector gradiente y evaluó la función objetivo 6 veces (`Nobj 6`) y la cantidad de variables superbásicas del modelo es igual a 1 (`NSB 1`).

f) la norma del vector gradiente reducido en el punto óptimo es aproximadamente 0 (`Norm RG 1.78E-15`).

En cuanto al archivo .lst, en el "equation listing" se vuelven a listar las ecuaciones del modelo. El listado muestra del lado izquierdo qué variables aparecen en cada ecuación, junto a sus coeficientes una vez realizadas todas las manipulaciones de datos. Si el coeficiente que acompaña la variable está entre paréntesis, la misma es no lineal. Para las ecuaciones no lineales GAMS realiza una linealización (paso previo necesario para la aplicación del vector gradiente), y el coeficiente que aparece entre paréntesis es la derivada de la función original con respecto a la variable en cuestión evaluada en el punto inicial de la iteración, que por defecto es 0.[22] En nuestro ejemplo particular la función objetivo es no lineal, y GAMS arroja el siguiente resultado:

```
Equation Listing    SOLVE non2 USING NLP FROM LINE 11

---- objetivo  =E=

objetivo..  phi + (0)*q + (0)*p =E= -1 ; (LHS = 0, INFES = 1 ***)

---- demanda  =E=

demanda..  2*q + p =E= 10 ; (LHS = 0, INFES = 10 ***)
```

En términos de lo dicho anteriormente, el (0) que acompaña a *q* en la función de demanda es $\frac{\partial \pi}{\partial q}$ evaluada en (P=0, Q=0).

Al lado de cada ecuación, aparece una sigla LHS=0 que nos indica que la expresión a la izquierda (Left Hand Side) original (antes de ser linealizadada) evaluada en el punto inicial (0,0) adopta el valor 0. Nótese que éste punto (0,0) no es factible, ya que viola la restricción de demanda. Los asteriscos (`***`) informan que se está violando la ecuación y el `INFES=1` nos indica la diferencia existente entre la expresión a la izquierda evaluada en el punto inicial y el término independiente a la derecha.

[21] Como veremos más adelante en el archivo .lst el marginal de *p* que debería ser 0 es EPS o sea aproximadamente 0.

[22] En caso de existir condiciones iniciales impuestas, GAMS tomará éste punto como punto inicial. Para mas detalles ver Brooke et al. (1998)

Las características básicas del “solve summary” son similares al caso lineal. Solo se incorpora alguna información sobre el solver en particular que ya analizamos más arriba al detenernos en el archivo .log.

```
                      S O L V E      S U M M A R Y

     MODEL   non2                OBJECTIVE  phi
     TYPE    NLP                 DIRECTION  MAXIMIZE
     SOLVER  MINOS5              FROM LINE  12

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      2 LOCALLY OPTIMAL
**** OBJECTIVE VALUE                7.3333

 RESOURCE USAGE, LIMIT          0.000      1000.000
 ITERATION COUNT, LIMIT         1         10000
 EVALUATION ERRORS              0             0

 MINOS5        Feb 14, 2001 WIN.M5.M5 19.8 108.043.039.WAT

     B. A. Murtagh, University of New South Wales
       and
     P. E. Gill,  W. Murray,  M. A. Saunders and M. H. Wright
     Systems Optimization Laboratory, Stanford University.


 Work space allocated           --    0.04 Mb

 EXIT -- OPTIMAL SOLUTION FOUND
 MAJOR ITNS, LIMIT             1     200
 FUNOBJ, FUNCON CALLS          6       0
 SUPERBASICS                   1
 INTERPRETER USAGE          0.00
 NORM RG / NORM PI     9.421E-16

                       LOWER     LEVEL     UPPER    MARGINAL

---- EQU objetivo      -1.000    -1.000    -1.000     1.000
---- EQU demanda       10.000    10.000    10.000     1.667

                       LOWER     LEVEL     UPPER    MARGINAL

---- VAR phi            -INF      7.333     +INF       .
---- VAR q              -INF      1.667     +INF       .
---- VAR p              -INF      6.667     +INF      EPS


**** REPORT SUMMARY :         0     NONOPT
                              0 INFEASIBLE
                              0  UNBOUNDED
                              0     ERRORS


EXECUTION TIME       =        0.000 SECONDS    0.7 Mb      WIN198-120
```

2.4.2) Optimización de función no lineal sujeta a restricciones no lineales:

El problema 1 se puede representar en GAMS de la siguiente manera:

```
Variables
Phi
P
Q
Cos
Equations
Objetivo
Demanda
Costo;
Objetivo.. phi =E= p*q - cos  ;
Demanda.. p=E=10-2*q ;
Costo..  cos=E=(q**2)+1 ;
q.l=0.1;
p.l=0.1;
MODEL non1 /ALL/ ;
SOLVE non1 using NLP maximizing phi;
```

El archivo .log correspondiente es:

```
MINOS5        Feb 14, 2001 WIN.M5.M5 19.8 108.043.039.WAT

 Work space allocated           --    0.04 Mb
 Reading data...
 Reading nonlinear code...

 Major Minor  Ninf  Sinf,Objective    Viol    RG    NSB  Ncon Penalty   Step
   1     0     1    0.00000000E+00  1.0E+00 0.0E+00  2     3  5.0E+01 1.0E+00
   2     5     0    3.60684845E+00  1.8E-01 2.1E-12  1    13  5.0E+01 1.0E+00
   3     2     0    5.63445754E+00  1.3E-01 7.1E-07  1    19  5.0E+01 1.0E+00
   4     2     0    6.74904670E+00  9.3E-02 1.6E-07  1    25  5.0E+01 1.0E+00
   5     2     0    7.22724601E+00  5.8E-02 2.0E-09  1    31  5.0E+01 1.0E+00
   6     2     0    7.37954690E+00  4.7E-02 3.1E-07  1    36  5.0E+00 1.0E+00
   7     1     0    7.33362324E+00  2.9E-04 2.9E-07  1    39  5.0E-01 1.0E+00
   8     1     0    7.33333333E+00  5.3E-13 1.1E-10  1    41  5.0E-02 1.0E+00
   9     0     0    7.33333333E+00  0.0E+00 1.1E-10  1    42  0.0E+00 1.0E+00

 EXIT -- OPTIMAL SOLUTION FOUND

 Major, Minor itns              9          15
 Objective function       7.3333333333333E+00
 FUNOBJ, FUNCON calls          42          42
 Superbasics, Norm RG           1    1.05E-10
 Degenerate steps               0        0.00
 Norm X,      Norm PI    9.06E+00    2.52E+00
 Norm X,      Norm PI    1.02E+01    2.12E+00  (unscaled)
 Constraint violation    0.00E+00    0.00E+00
--- Restarting execution
--- NONLINEAL.GMS(16) 0 Mb
--- Reading solution for model nonl
*** Status: Normal completion
```

Nuevamente Gams nos provee información acerca de la ejecución del solver. Mucha de esta información ya fue comentada en el caso de optimización no lineal con restricciones lineales.

a) Se listan las iteraciones "mayores"(`Major`) y correspondientes iteraciones "menores" (`Minor`) relacionadas con la resolución del subproblema por el método gradiente.

b) Se hace referencia nuevamente al número de restricciones que se están violando (`Ninf`), al valor asociado de la variable objetivo (`Sinf,Objective)` y al numero de variables superbasicas (`NSB`) para cada iteración.

c) Se indica además la máxima violación de la restricción no lineal (`Viol`) y se provee el valor del vector gradiente reducido para cada iteración (`RG`). Asimismo se indica la cantidad de veces que Minos5 evaluó la restricción no lineal y sus de derivadas en forma acumulativa (`Ncon`) y además se indica el valor del multiplicador de penalidad adoptado en cada iteración (`Penalty`). Por último, se proporciona información referente al tamaño del paso tomado en la dirección sugerida por el vector gradiente reducido para cada iteración (`step`).

Resulta interesante prestar atención a los sucesivos valores adoptados por el parámetro de penalidad. Inicialmente se asigna un valor $\delta = 50$. Este valor no fue cambiado inicialmente ya que las sucesivas iteraciones mayores mostraban una convergencia a 0 de la violación de la restricción no lineal. Cuando la violación llegó a 0 aproximadamente, δ pasó a ser 0.5 y luego 0.05, y ya dejó de ser necesario corregir la dirección del vector gradiente. Como era de esperar, en el máximo el vector gradiente reducido tiende a 0.

En cuanto al "equation listing" del archivo .lst tenemos:

```
Equation Listing    SOLVE nonl USING NLP FROM LINE 16

---- objetivo  =E=

objetivo..  phi - (0.1)*p - (0.1)*q + cos =E= 0 ; (LHS = -0.01, INFES =
0.01 ***)

---- demanda  =E=

demanda..  p + 2*q =E= 10 ; (LHS = 0.3, INFES = 9.7 ***)

---- costo  =E=

costo..  - (0.2)*q + cos =E= 1 ; (LHS = -0.01, INFES = 1.01 ***)
```

A diferencia del ejemplo anterior, hemos impuesto condiciones iniciales a la ejecución del solver, a saber : `q.l=0.1;p.l=0.1.` Es decir hemos decidido que el solver tome como punto inicial ya no el (P=0, Q=0) que vendría por default , sino el (0.1, 0.1). Nuevamente, las ecuaciones no lineales han sido linealizadas y valorizadas en el punto inicial. De ésta manera, por ejemplo, el coeficiente (0.2) que acompaña a *q* en la función de costo resulta

de $\frac{\partial C}{\partial q} = 2q$ evaluada en q = 0.1. Las siglas `LHS` e `INFES` tienen la misma interpretación que en el ejemplo anterior. Por último, el "solve summary" en éste caso es:

```
                     S O L V E      S U M M A R Y

     MODEL   nonl                OBJECTIVE  phi
     TYPE    NLP                 DIRECTION  MAXIMIZE
     SOLVER  MINOS5              FROM LINE  16

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      2 LOCALLY OPTIMAL
**** OBJECTIVE VALUE                7.3333

 RESOURCE USAGE, LIMIT          0.051      1000.000
 ITERATION COUNT, LIMIT        15         10000
 EVALUATION ERRORS              0             0

 MINOS5        Feb 14, 2001 WIN.M5.M5 19.8 108.043.039.WAT

     B. A. Murtagh, University of New South Wales
       and
     P. E. Gill,  W. Murray,  M. A. Saunders and M. H. Wright
     Systems Optimization Laboratory, Stanford University.

 Work space allocated           --    0.04 Mb

 EXIT -- OPTIMAL SOLUTION FOUND
 MAJOR ITNS, LIMIT             9     200
 FUNOBJ, FUNCON CALLS         42      42
 SUPERBASICS                   1
 INTERPRETER USAGE          0.00
 NORM RG / NORM PI     4.966E-11

                       LOWER     LEVEL     UPPER    MARGINAL
---- EQU objetivo        .         .         .        1.000
---- EQU demanda       10.000    10.000    10.000     1.667
---- EQU costo          1.000     1.000     1.000    -1.000

                       LOWER     LEVEL     UPPER    MARGINAL
---- VAR phi            -INF      7.333     +INF        .
---- VAR p              -INF      6.667     +INF       EPS
---- VAR q              -INF      1.667     +INF        .
---- VAR cos            -INF      3.778     +INF        .

**** REPORT SUMMARY :         0     NONOPT
                              0 INFEASIBLE
                              0  UNBOUNDED
                              0     ERRORS


EXECUTION TIME       =        0.000 SECONDS    0.7 Mb      WIN198-120
```

2.4 3) Estrategias para obtener convergencia:

En los problemas de optimización no lineal de cierta complejidad, no es infrecuente que resulte dificultoso para el solver el converger a una solución. En tales casos GAMS arrojará un mensaje del tipo:

```
LOCALLY INFEASABLE, ó
INTERMEDIATE INFEASABLE, ó
INTERMEDIATE NONOPTIMAL.
```

Existen sin embargo algunas estrategias que permiten, en muchos casos, arribar a una solución. Las mismas tienen que ver con la reformulación de la representación del problema, del cambio de condiciones iniciales, del numero máximo de iteraciones, o de algún parámetro específico del solver. En lo que sigue presentamos algunas de estas alternativas.

a) *Modificación de la forma de Representación del Problema*: Del sencillo ejemplo de la sección anterior queda claro que existen muchas formas alternativas de representar un mismo problema. Estas formas se multiplican a medida que el problema crece en complejidad. Asimismo, resulta evidente que el modo que el solver tiene de encarar cada formulación es distinta. Sucede a menudo, aún con problemas sencillos, que la resolución exitosa de un problema depende sensiblemente de su formulación. Usualmente se suele sugerir que uno presente el problema en la forma más "compacta" posible y, si es posible, colapsando las restricciones no lineales dentro de la función objetivo. Esto se debe a que los métodos de optimización sin restricciones o de optimización con restricciones lineales tienen un mejor comportamiento que, por ejemplo, el método de "penalidades" expuesto anteriormente: garantizan una mejor convergencia al resultado y son menos demandantes en términos de recursos computacionales.

b) *Modificación de condiciones iniciales*: A veces suele ser necesario modificar las condiciones iniciales sobre las que el solver empieza a iterar a fin de obtener convergencia. Las condiciones iniciales se fijan asignando un "nivel" inicial a la las variables. Por ejemplo en la formulación del problema de maximización de la sección 2.3.2, para evitar que el solver comience a iterar en el punto (0,0), se fijo un valor inicial de las variables:

```
q.l=0.1;
p.l=0.1;
```

c) *Número máximo de iteraciones*: Puede suceder asimismo que el numero de iteraciones máximas permitidas sea insuficiente para permitir al solver converger al punto estacionario. Para permitir más iteraciones deberá introducirse el comando Option iterlim = N° antes del comando **Solve**. Por ejemplo si deseamos que el numero máximo de iteraciones sea 2000 (por default son 1000) deberemos realizar la siguiente modificación:

```
MODEL non1 /ALL/ ;
Option iterlim = 2000;
SOLVE non1 using NLP maximizing phi;
```

d) *Modificación de Parámetros del Algoritmo*: En el caso de que sea muy difícil modificar la presentación el modelo y que su resolución presente dificultades, siempre existe la posibilidad de modificar los parámetros específicos del Solver. GAMS ofrece esta posibilidad creando un archivo de extensión .opt. El nombre de éste archivo debe ser el nombre del solver cuyos parámetros se desea modificar y debe estar en el directorio de trabajo. En nuestro caso se deberá crear el archivo Minos5.opt. Asimismo se deberá insertar una línea de comando en el archivo .gms para activar la modificación luego de definir el nombre del modelo y antes del comando Solve. En nuestro ejemplo particular:

```
MODEL non1 /ALL/ ;
non1.optfile = 1 ;
SOLVE non1 using NLP maximizing phi;
```

Dentro del archivo .opt deben ingresarse los distintas opciones de parámetros. Estas opciones suelen ser bastante numerosas para los solvers de optimización no lineal. Para un detalle de las opciones de cada solver se deberá recurrir al manual correspondiente. En particular, para el Minos5 veremos solo dos opciones:

1) Modificación del margen de tolerancia $|\Delta x| \leq \varepsilon$ *:* Como fuera mencionado anteriormente el solver se detendrá cuando el vector gradiente sea lo "suficientemente cercano"a 0. La decisión de que es lo "suficientemente cercano" puede ser modificada a través del margen de tolerancia ε. Para esto deberá insertarse la siguiente línea en el archivo .opt:

Optimality Tolerance N°

2) Modificación del parámetro de Penalidad inicial δ *:* Es posible que si se asigna una valor δ elevado la tasa de convergencia puede ser innecesariamente lenta. Para agilizar el cálculo se puede asignar un valor de δ moderado utilizando el comando:

Penalty Parameter N°

Para ejemplificar lo anterior se puede a reducir el margen de tolerancia de 1.0E-6 que viene por default y pondremos un margen más laxo de 1.0E-1. Asimismo reduciremos el valor del multiplicador de penalidad a 0.05 lo que generará una convergencia más rápida. El archivo Minos5.opt quedaría de la siguiente forma:

```
*minos5 option file
penalty Parameter 0.05
optimality tolerance 1.0E-1
```

Resolviendo nuevamente el ejercicio de la sección 2.3.2 podemos ver en el archivo .log los efectos de las modificaciones

```
MINOS5        Feb 14, 2001 WIN.M5.M5 19.8 108.043.039.WAT

 Using supplied options file
 > *minos5 option file
 > penalty Parameter 0.05
 > optimality tolerance 1.0E-1
 Work space allocated           --    0.04 Mb
 Reading data...
 Reading nonlinear code...

 Major Minor  Ninf  Sinf,Objective    Viol    RG    NSB  Ncon Penalty    Step
     1     0     1  0.00000000E+00 1.0E+00 0.0E+00    2     3 5.0E-02 1.0E+00
     2     2     0  1.09132255E+01 4.5E+00 2.8E-02    1     6 5.0E-02 1.0E+00
     3     1     0  7.64142180E+00 3.1E-01 5.3E-04    1     9 5.0E-02 1.0E+00
     4     0     0  7.33331187E+00 0.0E+00 8.0E-03    1    10 5.0E-02 1.0E+00
     5     0     0  7.33331187E+00 0.0E+00 8.0E-03    1    11 0.0E+00 1.0E+00

 EXIT -- OPTIMAL SOLUTION FOUND

 Major, Minor itns              5           3
 Objective function       7.3333118696618E+00
 FUNOBJ, FUNCON calls          11          11
 Superbasics, Norm RG           1    8.02E-03
 Degenerate steps               0        0.00
 Norm X,      Norm PI    9.06E+00    2.51E+00
 Norm X,      Norm PI    1.02E+01    2.11E+00  (unscaled)
 Constraint violation    0.00E+00    0.00E+00
--- Restarting execution
--- NONLINEAL.GMS(18) 0 Mb
--- Reading solution for model nonl
```

Claramente, dada la sencillez del problema la función objetivo no ha cambiado mucho. Sin embargo, se puede apreciar el cambio generado por un vector gradiente más alejado del 0. En cuanto al efecto de la menor penalidad, este ha redundado en un número menor de iteraciones (de 9 a 5).

Puede darse el caso de que no podamos obtener convergencia, es decir, una solución óptima, aún cuando reformulemos la representación del problema,[23] modifiquemos las condiciones iniciales, aumentemos el número de iteraciones, o modifiquemos parámetros específicos del solver. Y ello puede ser así porque simplemente el problema no admite solución. En tales casos es recomendable verificar, de ser factible, si se cumplen las condiciones de Kuhn-Tucker[24] de modo de dilucidar si existe una solución al problema planteado.

[23] Por ejemplo, tal como fue expuesto más arriba, incorporando las restricciones dentro de la función objetivo
[24] ver nota al pie 10

Bibliografía

Armitano, O., J. Edelman y U. Garcia Palomares (1985) *Programación no lineal*, Editorial Limusa.

Bazaraa, M., H. Sherali y C.M. Shetty (1995) *Nonlineal Programming, Theory and Algorithms* Segunda Edición , John Wiley & Son.

Brooke A., D. Kendrik, A Meeraus y R. Raman, (1998) Gams: a user´s guide, Gams Development Corporation (www.gams.com).

Chiang, A. (1987), *Metodos Fundamentales de Economía Matemática,* Mc Graw-Hill.

Gams - The solver Mannuals (2001), Gams Development Corporation (www.gams.com).

Henderson, J. y R. Quant (1985), *Teoría Microeconomica: Una aproximación matemática*, Editorial Ariel.

Judd, K. (1998), *Numerical Methods in Economics,* The MIT Press.

Mercado P. R., (2002), *Optimización Dinámica Restringida en Economía: Métodos Matemáticos e Implementación en el General Algebraic Modeling System.* Documento de Trabajo PIEC-DEyF-UADE.

Rardin, R. (1998), *Optimization in Operations Resarch,* Prentice Hall.

Silberberg E. y W. Suen (2001), *The Structure of Economics. A Mathematical Analysis,* Mc Graw-Hill.

Simon, C.y L. Blume (1994), *Mathematics for Economists,* WW Norton & Company.